TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Info technology
Department of Computer Science
Network Software

Joosep Simm

WAV30LT

AI System for Online Poker

Diploma Thesis

Supervisor: **Tanel Tammet**
Professor

Tallinn 2007

Declaration

Herewith I declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors are used only with a reference to the source. The thesis has not been submitted for any degree or examination in any other university.

Date:

Author: Signature:

Task of the thesis

Topic of the thesis:

AI System for playing poker in Internet environment
(in English)

Pokkeri tehisintellekt mängimiseks Internetis
(in Estonian)

Initiator of the topic: Joosep Simm

Aims: Creation of simple learning poker AI system for strategy of honest play and for strategy of cooperation play with other similar AI systems.

Expected results: A reasonable poker playing AI system. Cooperation with other similar AI systems should increase the total profit of cooperating players.

Problems to be solved: Developing and implementing poker AI system for usual and for cooperation strategy. Testing against various opponents.

Additional requirements:

Date:

Student: Joosep Simm Signature:

Supervisor: Tanel Tammet Signature:

Approver: Signature:

Joosep Simm

WAV30LT

AI System for Online Poker

Diploma Thesis

Abstract

The goals of the thesis are creation of a simple learning poker playing AI system that is also capable of cooperating unfairly with similar AI systems.

The main problems addressed in this thesis are developing the poker playing AI system that can learn opponents' strategy and adjust its play accordingly, also creating the unfair cooperation strategy between the similar AIsystems and thoroughly testing the performance of both strategies.

The result of the thesis is AI system for playing poker, learning opponent strategies and can improve its profit by using unfair cooperation strategy.

The thesis is in English and contains 45 pages of text, 6 chapters, 3 figures, 17 tables and 5 formulas.

Contents

Introduction

Computing science has received good response to chess, checkers, backgammon and Othello programs that beat the best human players. While these games can be said to be solved, there are games that are still mastered by humans. These games include poker and bridge. In the board games, all players have complete information about game situation; these are called *perfect information* games. In contrast in poker the opponent cards are hidden and also cards that will be revealed only in later betting rounds and such games are called *imperfect information* games.

Poker has been studied by mathematicians and economists, though simplified versions of poker was used. John von Neumann and John Nash illustrated the fundamental principles of game theory [4, 5 and 6] in poker.

Significant research has also been done by computing science in solving poker game. Team lead by Jonathan Schaeffer has identified the key aspects needed for world-class play [1, 2]. Koller and Pfeffer have created a system for creating game theoretic optimal solution for imperfect games, system called *Gala* [3]. The *Gala* system solves the game tree in polynomial time, but its performance is optimal. This is a great contribution for solving imperfect information games, but future work is needed to use in full-scale poker. Creating the game tree for full-scale poker is still too large to use Gala system.

University of Alberta has shown very good performance in playing 2 player *limit poker*. Last match between humans and UA poker program was in 2007 August [7]. Their program performed well and gained only a marginal loss.

The thesis is attempting to create a poker AI system for *no-limit poker*. One of the main tasks for the thesis is to investigate the possibilities of cooperation between players in poker. This poker AI system is called Jsbot, which can play in cooperation is other Jsbots

and without cooperation The paper will explain and demonstrate Jsbot AI system and its basic cooperation possibilities. The performance of Jsbot is tested against other publicly available *no-limit poker* AI systems. The testing was performed in 16 different environments, with total of 160 000 games of poker played. Future work is needed for the program to play world-class poker and to enhance profit from the cooperation between players.

Section 2 of this paper explains the rules of Texas Hold'em poker and explains basic strategy of poker. Principles of cooperation in poker are described in Section 3. Section 4 will describe algorithms used in poker programs in the past and will explain my poker algorithm in depth. Testing performance of a poker program is quite difficult, thus different methods are described in Section 5. The results of Jsbot AI system tests are described in Section 6. Section 7 discusses the future possible enchantment for the AI system demonstrated in this paper, including cooperation possibilities.

1 Poker – Texas Hold'em

The thesis has chosen to study the game of Texas Hold'em *no-limit*, the variation used to determine the world champion in the annual World Series of Poker. No-limit Hold'em is considered to be the most popular poker variation played in the world. This is especially true from the start of 21st century, when Texas Hold'em experienced a surge in popularity worldwide [8]. Many observers attribute this growth to the synergy of four factors:

1. The invention of online poker
2. The game's appearance in film and on television
3. The appearance of the television commercials advertising online card rooms
4. The 2003 World Series of Poker championship victory by online qualifier Chris Moneymaker.

The thesis will assume the reader is familiar with the ranking of poker hands (if not, many good explanation can be found on the Internet). Throughout the paper, italics are used for common poker terms, which are defined in the poker glossary, which is brought out in appendix 2.

1.2 Rules of play

Texas Hold'em poker game begins with each player being dealt two cards face down from a standard 52 card deck with no jokers, like most poker games. These cards are the player's *hole* or *pocket cards*. These are the only cards each player will receive individually, and they will only (possibly) be revealed at the showdown, making Texas hold'em a closed poker game. Closed poker game is a poker game in which no cards held by individual players are visible to any other player before the *showdown.*

Dealing hole cards is followed by posting two forced bets called *small-blind* and *big-blind.* The *hand* begins with a *pre-flop* betting round, beginning with the player to the left of the

big blind (or the player to the left of the dealer, if no *blinds* are used) and continuing clockwise. A round of betting continues until every player has either *folded*, put in all of their money, or matched the amount put in by all other active players. Note that the *blinds* are considered "live" in the *pre-flop* betting round, meaning that they contribute to the amount that the blind player must contribute, and that, if all players *call* around to the player in the *big blind* position, that player may either *check* or *raise*.

After the *pre-flop* betting round, assuming there remain at least two players taking part in the hand, the *dealer* deals a *flop*, three *face-up community cards*. The *flop* is followed by a second betting round. This and all subsequent betting rounds begin with the player to the *dealer's* left and continue clockwise.

After the *flop* betting round ends, a single *community card* (called the *turn* or *fourth street*) is dealt, followed by a third betting round. A final single *community card* (called the *river* or *fifth street*) is then dealt, followed by a fourth betting round and the *showdown*, if necessary.[9]

1.3 Basic Poker strategy

There are numerous articles and books about poker strategy. The best books accepted by publicity are from David Sklansky and Mason Malmuth [10, 11, 12 and 13]. The fundamental theorem of poker is a principle first articulated by David Sklansky that he believes expresses the essential nature of poker as a game of decision-making in the face of incomplete information.

„*Every time you play a hand differently from the way you would have played it if you could see all your opponents cards, they gain; and every time you play your hand the same way you would have played it if you could see all their cards, they lose. Conversely, every time opponents play their hands differently from the way they would have if they could see all your cards, you gain; and every time they play their hands the same way they would have played if they could see all your cards, you lose.*"[12]

This theory is perfectly true, but this alone does not give guidelines how to make good poker playing program. The thesis will now give an illustrative example of usual thinking process of a poker player. This example is quite simple for easy following. Poker situations can be much more complicated than the following example.

The game is $0.5-$1 no-limit Texas Hold'em with ten players. Each player *stack* size is $100, effectively 100 *big blinds* worth of *chips*. Let's call the player we are looking at Alice. Alice is on *button*, meaning she will be the last to act in each betting round except *pre-flop* (*big-blind* is last to act in the *pre-flop* betting round). *Blinds* are posted and first 5 players fold. 6th player calls, I'll refer to him as MP (*middle position*) and next player folds.

Its Alice's turn, she holds Tc and 9c. (See Appendix B for explanation of poker card abbreviations). She knows that the hand is good drawing hand, with possibility to get *straight* or *flush*. Calling with this hand would make sense if the players' in *blinds* would not make a raise. From earlier play she knows the blinds would only make a raise if they

hold good cards, in other words they are not bluffing. A possibility to *raise* cannot be disqualified too. If she would *raise* she might win all the money immediately currently in the pot or can *bluff* in later betting rounds, because she showed *strength* in *pre-flop*. Though when making the *raise* she is open to *re-raise*, which would mean she should *fold*, because her cards are only good if she *hits the flop*. So she calls and *blinds* also *call* and *check*.

The flop is dealt – 2h, 4d, Td. Alice has the *top-pair*, making her hand quite good. Though there are quite a few hands better than hers: 42, T2, T4, 22, 44, any ten with higher *kicker* and all higher *pocket-pairs*. As no-one raised before the flop, then probability of someone holding higher *pocket-pair* than pair of tens is very low, because players usually raise with high *pocket-pairs* to get rid of players with smaller cards. From earlier play she has put MP on tight-player, so he should be holding high-cards or also drawing cards like Alice. Both blinds can hold about anything, with exception of high cards, so the flop could have been good for them.

Player on small blind (I'll refer as SB) bets $4 (the amount in the pot). Player on *big blind* and MP both *fold*. Alice has now choice to *fold*, *call* or make a *raise*. Raise has to be at least $4 or more. From earlier plays she knows that SB likes to *slow-play*, so his possible *pocket-cards* should not be too strong. Alice excludes *2 pair* and *3 of a kind* possibilities. As she knows that SB is a tight player, then he should have at least a pair or a draw. Effectively leaving the SB on *pair of* tens, *pair of* fours or on *a flush draw* or *a straight draw* making a *semi-bluff*. If SB is *semi-bluffing*, then correct move would be to *re-raise*, making the extra cards too expensive for SB. On the other hand, if SB has ten with a higher *kicker* he would *call* Alice. If Alice would decide to *call*, then she should decide again next round if SB *bets*. On the other hand if SB checks in the next round, it would mean he is on flush-draw and Alice can *raise* then. So Alice decides to *call*.

Turn is dealt and its 8c. This card doesn't help either of players. SB checks and it is Alice's turn to act. If SB would have had ten with good *kicker*, then he should have *bet*, to make

buying *flush* or *straight* more expensive for Alice. Alice excludes ten with higher *kicker* and makes a *pot* size *raise* herself. SB thinks for a while and *folds*.

A lot of information was shared in this game. For instance SB most probably had a *draw*, *top pair* with bad *kicker* or *second pair*. He doesn't raise (bluff) pre-flop with bad hand, when facing 3 opponents. He makes *pot* size *bet* after *flop* while being *under the gun* and holding a moderate *hand*. When his *semi-bluff* has been *called* he does not make another *bet* on the next betting round. Some conclusions can be made for other players as well. MP did not get reasonable *hand* after the *flop* was dealt and decided to *fold* whatever he had. He was either afraid of SB bet or afraid of Alice *calling* or *betting*, maybe MP was afraid of both. Alice can make even conclusion about her *table image*, when her call was taken seriously. She has table image of betting for value, meaning that bluffs in right circumstances are profitable for her. For example, she could have made the same play without catching the top pair with*flop* and would have won the moneyanyway.

In conclusion there are many aspects to evaluate before making the correct move in poker game. The best poker players know what information to gather and how to analyze it. Weaker players might gather all the information and analyze it incorrectly or not gather enough important information.

1.4 Requirements for a winning poker program

Previous studies on poker have brought out several aspects for world-class poker player [1, 2]. The focus of this paper is not on poker player in general, but on computer program perspective. The thesis is interested in the requirements that can be computed in real time with computer program. The requirements that will be brought out are used in Jsbot. These ideas covered here are surely not ideal and can be improved. The basic ideas are tested using basic algorithms to prove their importance.

Opponent modelling is by far the most important factor in no-limit poker, because enormous bluffs can be made and also caught. The profitability from making or catching bluffs depends on the frequency they are made. For example, compare two players. One is going *all-in* every 2nd game, the other goes all-in every 100th game. From those observations its easy to deduct, that first player goes all-in with 50% of the better hands or worse, while the second player is risking all his money only with 1% of best cards. It is easy to see that calling the first all-in player with 10% of best hands should be profitable in long run and calling the 2nd player all-in with 10% of best hands is not profitable. This is not true only with *all-ins*, but same statistics can be made for all bet sizes.

Another important factor in opponent modelling is the situation where players make these plays. For an example compare a player who raises when opposing 1 opponent and all other 8 opponents have *folded* or a player *raises* when he is first to act and 9 opponents are still holding their cards. In first situation the probability of the last opponent to have good *hand* is much smaller than in the 2nd example. The situation of game state is a mix of attributes: players in game, *pot size*, players *stack size*, *bets* made in this game, history of past games etc. In depth explanation of Jsbot player modelling will be provided in section 4.2.4.

Hand strength shows the hand strength compared to all other possible hands. This is used to calculate expected value in showdown situations. Also hand strength contains information about hand improving (i.e. *flush or straight draw*).

Betting strategy is strategy how to bet when you have processed all the information. I.e. slow-playing a hand, when you expect the opponent to make a raise or raising with weak hand when expecting opponent to fold. The most basic strategy is to bet for value, meaning that you bet when the expected value of your hand is positive.

These are the most important requirements for playing winning poker. For tuning the poker program for world-class play, some other aspects should be included. For example randomization of the play should be included to hide player strategy. This has been showed by Kuhn in his work with simplified poker [15], that randomization is required for optimal strategy in imperfect games.

2 Cooperation in Poker

Cooperation in poker is usually called collusion, which is when two players share a secret strategy. I mean under cooperation also sharing pocket-cards. Surprisingly it has not been studied before, most probably because it is not allowed in poker games and is called cheating. However, cooperating in online-poker nowadays is fairly easy. Two players can sit behind two computers next to each other and play in the same table. Online casinos prohibit it and have some tools against such activities, but with not great success. For examples online casinos are checking player IP addresses and do not allow two players from the same IP address to the same poker room. With VPN this check can be eliminated very easily.

The interesting aspect in studying cheating is knowing how much cheating affects the outcome of a poker play. I assume that traditional cheating is out of the question. By traditional I mean stacking the deck or peeking opponent cards. Though peeking at player and upcoming community cards cannot be out of the question in online poker as is described in [25]. This is because many online poker sites show their deck randomization algorithms to prove the correctness of dealing. From the algorithm, it is possible to see where the random seed is taken. After that, it is easy to recreate the algorithm in your own computer and after the cards are dealt, you can generate many close deals. This is especially true if no hidden information is used for generating random seed. For example if seed is taken directly from the poker server time. Then you can just generate cards for all close times. And if you know the *pocket cards* of one or two players, you can pick those seeds that match. Thus, you will know what cards your opponents might hold. If you have more information (after the flop is dealt or you know more players' *pocket cards*), then you can say with higher precision what cards opponents have and what *community cards* will be dealt next.

The thesis is more interested in sharing the pocket-cards of cooperating players and creating strategy from knowing that information. There are three benefits that come from the increased information and shared strategy:

1. Calculating hand strength and opponent cards increase in accuracy. Probably Omaha Hold'em would gain bigger benefit from the information, as 4 cards are dealt to each player. Otherwise the game is similar to Texas Hold'em.

2. Cooperating players don't play against each other, especially when one of them is holding superior cards compared to the other. This is called *soft play*.

3. Lurking „victims" in the pot with small bets, when one of the cooperating players have very strong hand. It is not studied further in this paper.

3 Poker algorithms

Perfect information games have the best move in all game situations, while in poker the best single move cannot be always determined. Of course game theory applies to poker as well, but game theoretic solution gives us optimal strategy. In imperfect information games the optimal strategy is not the best one. For example consider a game of „rock-paper-scissors" where an optimal strategy is to just randomize your play. The outcome is that you will not lose to any opponents, but the option of winning is also cancelled.

The maximized strategy for poker is to model opponents and exploit their weaknesses. This would also mean that there are weaknesses in applying the maximized strategy, because it rarely is the same as optimal strategy. In an ideal world where two players have the same knowledge of poker, but differ from their strategy adoption speed, the one who adopts faster will win in poker.

While significant research has been made for limit poker, very few works has been done studying no-limit poker. Rickard Andersson has studied 2 player no-limit poker game, when *stack* sizes are 15 big blinds or less [16]. He showed successfully, how to use game theoretic approach in such games, without opponent modelling. Unfortunately his algorithm needs to be modified for poker with normal (~100 big blinds) stack sizes.

3.1 Poker Academy no-limit Poker algorithms

While there have been some poker program competitions, the best no-limit poker programs are not available for testing. For testing Jsbot bot the thesis used Poker Academy [24] poker programs for testing. Poker Academy will be introduced in more detail in section 5. All no-limit poker programs in Poker Academy are rule based, with no opponent modelling. Even though they are missing an important factor that all good no-limit human players have, they are playing solid strategies. Some algorithms made by University of

Alberta AI team and some strategies published by famous poker authors. Here is the list of poker algorithms I tested with Jsbot: AveryBot, ReRaiser, OddBot, TournyBot, Sklansky, PKRoomBot and Solid. These opponents' strategies are covered in more detail in section 5.3.

3.2 Jsbot algorithm

Jsbot algorithm models each opponent separately and makes most of its calculations based on that model. Algorithm uses also its hand strength and pot-odds in calculating expected profitability for each action, but these should be basics in every reasonable poker algorithm. The thesis will now describe the process of action choosing from the most generalized view and then explain each part of algorithm in detail.

For a quick summary, the Jsbot algorithm calculates expected value for a couple of possible actions and chooses the most profitable action. Expected value is calculated by estimating win percentage as the game would go to showdown and only this round betting is taken into account. Effectively *implied-odds* are not considered in this calculation. Also future opponent *raises* are not considered, thus it cannot estimate the value of *slow playing*. The strategy of Jsbot can be said to be betting for value. Value comes from winning at the *showdown* or from situations when all opponents *fold* and Jsbot takes the *pot* uncontested. A more detailed explanation of each sub-algorithm is explained in the following sections.

3.2.1 How Jsbot chooses the action

In no-limit poker the number of possibilities of the size of the raise is very big. Player can raise any amount of chips bigger than the minimum raise, where minimum raise is big-blind or the size of previous raise in this round. Hence, formula (0) is used to calculate reasonable bet sizes in order to look through all reasonable raises, while skipping raises that

do not make much difference in the game play. For example, consider a situation where 40 big blinds are in the *pot* and *betting* 40 or 50 *big blinds*. In one case the *pot* is increased by 100%, in the other case the *pot* is increased 110%. The difference is so small, that opponents have to make basically the same decision.

Jsbot looks through 7 options: *check/fold*, *call*, *raise-small*, *raise-medium*, *raise-big*, *raise-huge*, *raise-all-in*. These seven options define the bet size used for other calculations. More bet options can be viewed, but it would increase the calculation time. The 7 categories are chosen by the authors' knowledge of poker and are not proved to be correct.

Check/fold is obvious. *Fold* is always an option and *bet size* is 0. *Check bet size* is 0, but it is not an option when a *raise* has already been made. *Call bet size* is the amount needed to *bet* to level the current raise.

For other *bet sizes* the real *bet size* is calculated with the following formula:

$$BigBlindsToBet = (BETSIZE * potBeforeAction) + amountToCall + bbsInPot \qquad (0)$$

Where

potBeforeAction is pot size in big blinds before the bot action;
amountToCall is amount needed for calling in big blinds;
bbsInPot is amount already put into pot by Jsbot in big blinds;
BETSIZE is percentage from the raise class.

Raise classes are the following:

Small – 25%
Medium – 50%
Big – 100%
Huge – 175%
All-in – 400%

For each of the seven options a number of variables are calculated in the following order:

1. estimated opponents
2. estimated pot size at the end of this betting round
3. Jsbot win percentage
4. profit

An explanation is in the Table 1, where you can see variables of Jsbot action choosing in a random point in poker game. The game *round* is *pre-flop* and Solid1 has called the *big blind*. This explains why he is considered the opponent when Jsbot would *raise*. He is the only opponent who has shown its *hand* strength. Jsbot has not made any decision for other players yet and considers their *hand* to be totally random. As you can see, the win percentage when playing against Solid1 changes little bit, but the magnitude is the same. At the same time when opposing more opponents, the magnitude of the winning percentage drops quite a bit.

Table 1. Jsbot action list

Action	**Bet Size (big blinds)**	**Estimated Opponents (playing probability and opponents)**	**Pot Size (big blinds)**	**Win percentage**	**Profit (big blinds)**
Check/fold	-	-	-	-	**0**
Call	1	4 (TournyBot1, Sklansky2, Solid1, Jsbot2)	5.5	20.95%	**0.15**
Raise-small	2	0.76 (Solid1)	5.5	36.80%	**0.88**
Raise-medium	3	1.56 (Solid1, TournyBot1)	10.5	26.25%	**-0.24**
Raise-big	5	0.76 (Solid1)	11.5	35.70%	**-0.51**
Raise-huge	7	0.76 (Solid1)	15.5	35.45%	**-1.44**
Raise-all-in	15	0.76 (Solid1)	31.5	38.60%	**-3.46**

Source: Example data from a random test game

After all variables of actions are calculated, the action with the biggest profit is chosen. In the given example small raise is made with *bet size* of 2. If all other action profits would be negative, then *check/fold* is the chosen. As said before, this kind of logics does not consider *slow playing*. Against some opponents *slow playing* a strong hand would be much more profitable than just *betting* against them.

3.2.2 Calculating number of players

For each player Jsbot calculates the playing probability after Jsbot has made its action. Each player has estimated number of hands before the Jsbot action and after the action. Playing probability of each player is defined as ratio between estimated hands number after the bot action and estimated hands number before bots' action.

It is assumed that the player will continue playing – *calling* or *raising* – when the player has one of the estimated hands after bots' action and otherwise *fold*. The logics for calculating number of players is quite intuitive, but it is only correct if the opponents hands are estimated correctly. While the opponent hand estimation cannot be very precise, because of the hidden information, these logics still work. Opponents who have *called* or *raised* before Jsbot action, are the most probable opponents. This is because their number of possible *hands* before Jsbot action is already smaller than for other players. For detailed explanation how player *hands* are estimated see section 3.2.3

3.2.3 Estimating opponent hand

Jsbot uses a straightforward algorithm for estimating opponent hand. It is assumed that players *bet* with better *hands* more than with weaker *hands*. While this is true for most players, this assumption can easily be used against Jsbot, *betting* with weaker *hands* and *checking/calling* with better *hands*. Is order to overcome this weakness, players' playing

patterns should be analyzed to find situations where players are *slow playing* or *bluffing*. This is not a trivial task and is left to the future study in poker.

Each player has one possible 2-card *hand* from possible 1326 2-card *hands*. When a player makes an action, possible *hands* are estimated for that player. *Hands* are also estimated when Jsbot is looking through its possible actions. In this situation the opponent action is considered *call* to the Jsbot *bet*. It is also assumed that a player *raises* the highest with the best *hand*; r*aises* little bit less with little bit worse *hand* and so on. For example, if a player is *raising* 10 big blinds 20% of the time, then he is holding 20% of the best *hands*. The assumption is wrong if a player *raises* with worse *hand*. Though, it is more profitable to play against a weaker *hand* in general.

Estimated *hands* for each player depend on their model, in other words on their playing history in the given game situation. Players' past actions for the given situation is searched and categorized by the *bet sizes*. Let's call *bet sizes* below current player *bet size out* and *bet sizes* same or higher *in*. The result of formula (0) shows the percentage of the hands the player should be holding. To get the estimated *hands*, all *hands* are ranked and the percentage is taken from the list from the top.

$$percentage = in/(in + out) \qquad (0)$$

Every time a player makes an action, then his estimated *hands* are updated accordingly. It does not give an exact possible hand list, but a rough estimate. To improve this estimation, *showdowns* and player playing patters should be analyzed. For example some players like to *slow play* very good *hands* and *raise* with medium *hands*.

3.2.4 Player modelling

An example of a very simple player modelling would be to count the players' *folds*, *calls* and *raises*. From that you can see that a player is *folding* X times, *calling* Y times and *raising* Z times. A simple conclusion might be that when that player is *raising*, then he is holding Z/(X+Y+Z) percentage of the best *hands*. While the statistics is true, it holds little value when analyzing exact game situation. To improve this model, more attributes from game situation should be considered. A simple example for improving the model would be to count the same statistics, but for two different situations. First situation is when someone has *bet* before that opponent; second situation is when no players have *bet* before that opponent. This improvement would give more accurate information about the player. The same logics are used for Jsbot player modelling, with more attributes considered for each game situation. Hence, the player actions' probabilities are saved in each game situation. This categorization results with a few records for each game situation and an average fold, call or raise percentage is never used.

Jsbot also categorizes player actions. There is a big difference if a player makes a minimum *bet* or a *bet* with size of 2 times the *pot*. The same is true for *calling* a *bet*. Each player action is divided into categories, similar to the way actions were categorized when choosing possible actions. Those actions are tied with each situation in game.

Attributes for categorizing the game situations are categorized into groups for faster searching and faster adoption to each player individual strategy. The categories are defined by the authors' expert knowledge about poker. If more categories would be used, the precision would be better, but learning would be slower. If fewer categories would be used, the precision would hinder, but learning would be faster. The following attributes are recorded for each game situation: *pot size, amount to call, players acted, players to act, last action this round* and *previous round last action*. Details about each attribute are brought below.

Pot size (in big blinds) before the player has made an action. Pot size is also divided into categories similar to bet sizes. The following categories are used to fasten the search and opponent modelling:

SMALL is when *pot* is equal or smaller than 6 *big blinds*.

MEDIUM is when *pot* is between 7 and 12 *big blinds*.

LARGE is when *pot* is between 13 and 20 *big blinds*.

HUGE is when *pot* is between 21 and 30 *big blinds*.

MAX is when *pot* is bigger than 30 *big blinds*.

Amount to call is the amount (*in big blinds*) player needs to *bet* in order to *call* the previous *bet*. This attribute is categorized similar to *bet sizes*, showing the percentage of the pot. For example, *calling* 10 *big blind raise pre-flop* with 2 *big blinds* in the *pot* requires a very good *hand* to be justified by *pot-odds*, but *calling* 10 *big blind raise* after *flop* can be done with much weaker hand when 40 *big blinds* are already in the *pot*. The categories used have been modified quite a bit in the process of testing Jsbot. In the beginning of testing the *Amount to call* showed total *big blinds* that had to be *called*. It came out quickly that percentage of the *pot* is more important compared to the absolute number of *big blinds* to *call*.

The following categories are used for *Amount to call*:

ZERO is when *bet to call* is 0, in other words a check.

SMALL is when *bet to call* is smaller or equal to 25% of the *pot size*.

MEDIUM is when *bet to call* is between 26% and 70% of the *pot size*.

BIG is when *bet to call* is between 71% and 120% of the *pot size*.

HUGE is when *bet to call* is between 121% and 250% of the *pot size*.

ALL-IN is when *bet to call* is bigger than 250% of the pot size.

Players acted shows the number of players acted and not *folded* in this betting round. The count starts from player who cannot make an action if all the other players either *fold* or *call* and ends with the current player. An illustrative example shows the *Players acted* calculation in Figure 1. The *Player acted* in this example for Jsbot is 1.

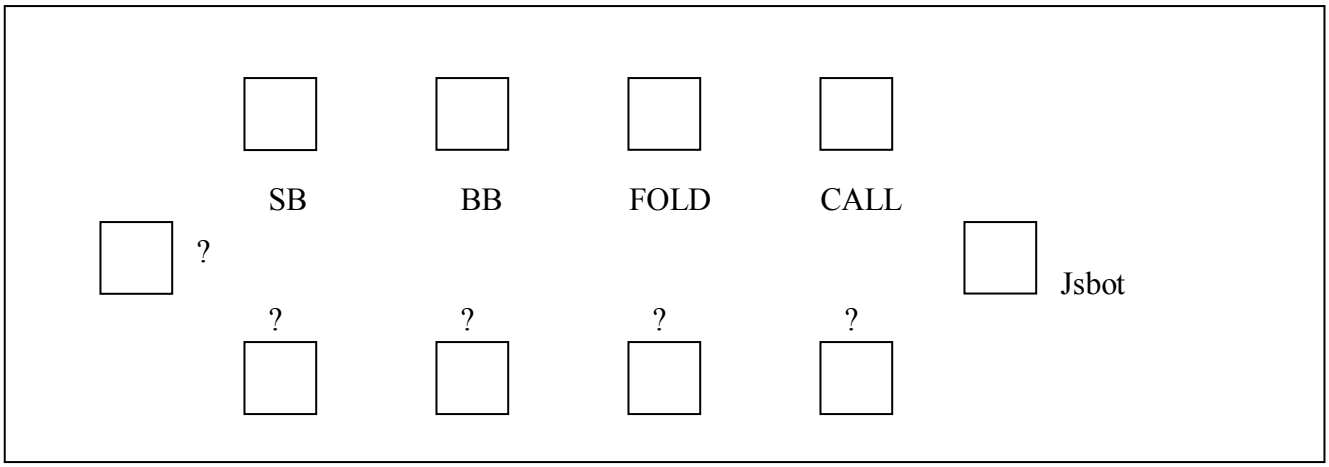

Figure 1. Example 10 player table: SB - small blind; BB - big blind; Fold - player has folded; Call - player has called; ? - not acted player

Players to act shows the number of players who can still make an action in this betting round. The count starts from the current player and ends with the player who cannot make an action if all the other players *fold* or *call*. An illustrative example shows the *Players to act* calculation in Figure 1. The *Player to act* in this example for Jsbot is 7.

Last action this round is the last player action made in this betting round.

Previous round last action is the last action the player made last betting round. This attribute is not used in *pre-flop* situations as there is no previous betting round.

While adding the data to a player history is easy, then searching important data from the player history is not trivial. Very many different game situations can be formed with those attributes and when searching for one particular game situation, usually only a few records exist. This is why Jsbot is using nearest neighbor search [21] and sums 25 or 15 nearest situations when player history is searched for. The number of nearest situations depends from the amount of data gathered for the player. The search can further be enhanced by searching until a certain number of records are found. Also forgetting older records is useful in case opponents' strategy should change. These two enhancements are not done for

Jsbot. Those changes would require measurement of the player model accuracy, in order to forget old data and search only those closest situations when accuracy remains optimal.

3.2.5 Estimating probable opponents

Estimating the most probable opponents Jsbot has to face is very intuitive. When Jsbot has calculated the playing probability for each player, then the sum of those probabilities gives a rough number of opponents the bot will be facing. The opponents with the highest playing probability are chosen.

In case Jsbots are cooperating with each other, they do not add other cooperating Jsbots to the estimated opponent list. This is done because the *pocket cards* of cooperating players are known and calculating win percentage against those *hands* does not give any information for playing against other opponents.

3.2.6 Calculating win percentage

Jsbot win percentage is calculated with Monte Carlo simulation. *Community cards* until *showdown* are dealt randomly and opponent *hands* are taken randomly from their estimated *hands* list. With each run Jsbot winning or losing is recorded. This process is repeated 1000 times to give rough estimate of win percentage. More simulations would give better estimate, but as other data is not precise, the estimate is accurate enough. The winning percentage with exactly the same input data differs ±3%, so the winning percentage accuracy is sufficient for calculating expected value of an action.

3.2.7 Calculating profit

Profit is calculated differently in two scenarios. One scenario is when the sum of opponents playing probability is over 100%, meaning there will be opponents left after this betting round. In this case formula (0) is used:

$$\mathrm{Pr}ofit = pot * winPercentage - \mathrm{cos}tOfAction \quad (0)$$

Where

pot is the estimated *pot* (in *big blinds*) after the estimated players have called

winPercentage is the Jsbot win percentage

costOfAction is the *bet* (in *big blinds*) Jsbot needs to make for this action

In case the sum of opponents playing probability is under 100%, there is a chance that opponents will *fold* and the game will not go to *showdown*. In this case win percentage of the *hand* is not important. Hence, profit consists of two parts: profit from opponents *folding* and profit from an opponent *calling*. For calculating these profits formulas (0) and (0) are used.

$$Call\ \mathrm{Pr}ofit = play\% * (pot * winPercentage - \mathrm{cos}tOfAction \quad (0)$$

$$Fold\ \mathrm{Pr}ofit = (1 - play\%) * potBeforeAction \quad (0)$$

Where

play% is the total playing probability of opponents

potBeforeAction is pot before action in big blinds

Other variables are the same as in the formula (0)

This profit calculation algorithm often tells Jsbot to go *all-in*, because such actions are very infrequent by players and *play%* is very small. For that reason *bluffing* is cut out by allowing minimum *CallProfit* to be -7 *big blinds*. Also *CallProfit* is multiplied by 2,

because this formula does not take into account next betting rounds and gives too optimistic results for hands with low win percentage.

3.3 Jsbot cooperation algorithm

Cooperation or collusion in poker is prohibited and is called cheating. If players are caught doing that, the players are removed from the game and possibly get ban from the casino they are caught at. How cooperation affects poker play and what are the benefits of it have not been studied before. If a cooperation strategy is played out well, then average profit for each cooperating player should be higher than without cooperation. As mistakes can be done during game play, mistakes can also be done cooperating with other players and profits decrease. The thesis is going to test very simple cooperation strategy.

Cooperating Jsbots share their *pocket-cards* with other cooperating Jsbots. This increases the accuracy of calculating win percentage and also gives better estimation for opponent possible *hands*. While the increase in accuracy is small, it should increase overall profit in the long run.

The second strategy cooperating Jsbots have is that they do not play against each other. This is achieved by not choosing cooperating Jsbots to the estimated opponents list. Some tests were made without this strategy, but it came out quickly that very wrong moves were made by Jsbots, because the exact information about other Jsbots *hands* gave reason to make high *raises*.

By applying those two strategies, the outcome of Jsbots strategy changed dramatically.
Main effect from the two strategies is that Jsbots started to play different strategy from the non-cooperating strategy. The cooperation strategy of Jsbots is the following. When an opponent is in the same *pot* with other cooperating Jsbots and raises against them, then the Jsbot with strongest *hand calls* or *raises* that player bet (of course only if *calling* or *raising* is justified). In some situations it is justified *calling* or *raising* by multiple Jsbots or by

none. The outcome of this strategy is that opponents face better hands and lose more often to Jsbots. The results of the cooperation strategy are seen in section 6.

4 Jsbot implementation

Jsbot was implemented in Java after considering different programming languages pros and cons. First of all logical and functional programming languages were excluded because the community of those is small compared to more popular object oriented languages and finding examples and help would be more difficult. The most popular object oriented languages are Java and different versions of C. While Java is known for its high use of memory and difficulties in optimizing the performance, then C requires very good understanding of memory management and implementation gets more complex because of those possibilities. So Java was chosen in hopes for easier implementation and with hope to avoid performance issues by using faster algorithms in general. Latest version (1.6) of Java was chosen, because the application is not meant for production usage and there is no legacy code that requires older versions of Java.

For logging the Apache logging package Log4j was chosen because of its ease to use and robustness.

Jsbot was implemented according to object oriented programming principles. Interfaces and classes were created for reusing and easily changing them if different logics were needed. The basic structure of classes of Jsbot is described in Figure 2. The arrows represent the usage of a class in the class where arrow is initiated from. *Utilities* and *Cards* are used by most of other classes, so arrows were not drawn to keep the figure clearer. Details about each package/class are below the Figure 2.

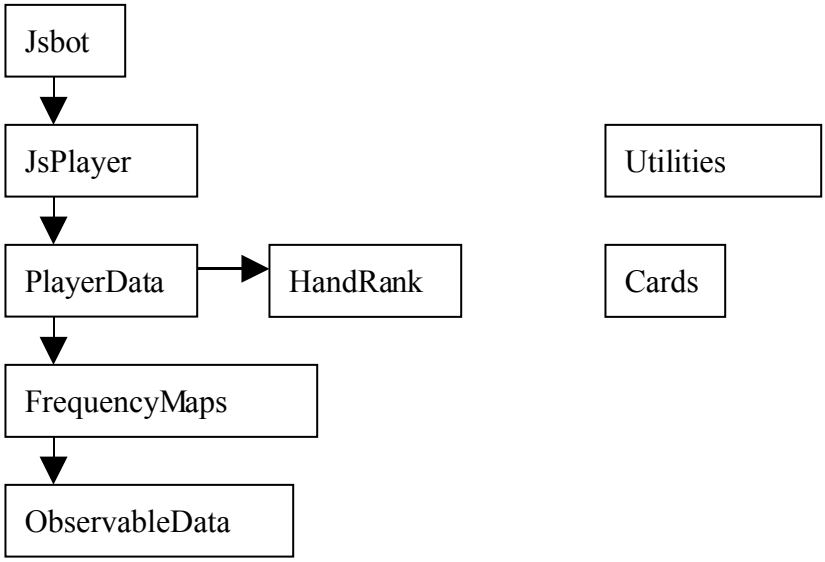

Figure 2. Jsbot class structure

Jsbot is so called "core class" which contains logics for choosing appropriate action depending of the calculations made. It contains list of all players and uses Utilities class for many calculations

JsPlayer is a general class for all players, containing their data and methods for saving and loading the particular player.

PlayerData contains the action frequencies for different situations and methods for getting different data from the frequencies. It contains also the hand rank object for particular player.

HandRank is an object that ranks hands in given situation. For *pre-flop* the hand ranks are pre-calculated for each number of players in the table. For *flop*, *turn* and *river* round hand rank is calculated in real time, because pre-calculation results in a huge database as the number of different *community cards* is very huge.

FrequencyMaps package contains the action frequencies for each *game situation* and the methods for searching the action frequencies from the database. *The game situation* is defined by the attributes that is considered in each game situation and is described in ObservableData package. For performance considerations the FrequencyMaps are held in memory as reading the amount of info from hard drive is very expensive (expensive in consideration of time spent).

ObservableData package contains the categories of attributes considered in each game situation and the definition of *game situations*. Distance between two *game situations* is also defined in ObservableData package. The distance is used in the nearest neighbor search.

Utilities package contains static methods for different tasks.

Cards package contains simple card objects and also a deck representation from 4 integers. The integer representation is used when calculating winning probability, because many operations are made during the calculation. A *deck* is shuffled, cards are taken and *hands* are compared to each other. Each integer represents 13 cards of a *flush* and all operations are done as bit operations to maximize the performance.

4.1 Performance of Algorithms

Many time consuming algorithms are used when Jsbot is calculating the profits for possible actions. The performance of those algorithms is critical to give a real time answer which action to choose. An average time it takes for Jsbot to choose its action is 3-4 seconds when opposing a full table of opponents. When playing against a single opponent, then approximately 1 second is needed to choose the action. The average time the most time consuming algorithms take to run are brought below.

Calculating winning probability takes from 250-750ms to compute. This depends from the number of opponents Jsbot is facing. With each Jsbot turn the winning percentage is computed 5-6 times, hence time consumed is very critical here and this is why special integer representation of a *deck* is used.

Player modelling can take a lot of time, because all playing histories of the players are stored. When looking at each possible action, Jsbot estimates the opponents possible *hands* based on the opponents' model. For example, opponent cards are estimated 40-48 times when looking at possible 5-6 future actions and opposing 8 opponents. For each possible *hand* estimation the playing history for certain game situation is searched from the frequency maps. Because game situations are described in high level of detail, then considering one situation is not enough and 15-25 closest situations are summed together. Because the time consumed is critical, the nearest neighbor search was chosen for doing this task, resulting in 10-30ms for estimating one player hand. At the early stages of Jsbot implementation a different algorithm was used where ~200ms for one hand estimation was wasted and totally ~8 seconds was required when opposing a full table of opponents.

Memory requirements are quite high due to holding player playing history in memory. The default memory java virtual machine (JVM) uses is 128MB. This was not sufficient to run 3 Jsbots in one game and over a gigabyte of memory in now allocated for JVM to test with multiple Jsbots.

4.2 Implementing cooperation

Jsbot cooperation requires cooperating players to share their pocket cards. As playing occurs only in one computer, the cooperation is not implemented to support playing in the net. This can easily be changed using server-client or peer-to-peer architecture. Currently each Jsbot writes the following data into a file:

1. cooperation boolean
2. unique game id
3. its *pocket cards*

Other Jsbots who are cooperating read the file and store the *pocket cards* that are from cooperating players who are in the same game with them. Java properties file format was chosen for this task, because methods for reading the data already exist in Java Properties object.

4.3 Implementing testing

For proper evaluation of Jsbot performance playing many games against many different opponents was needed. It is difficult to test against human opponents tens of thousand games, so existing poker algorithms was needed for this task. Poker program Poker Academy Pro 2.5 (PA) [24] was chosen as a *testbed* . PA is a program that runs under Windows operating system and consists of many poker algorithms. It also has Java API for connecting custom bots into the program, called Meerkat API, which can be downloaded from [22].

While PA has opponents and it was easy to connect Jsbot to the program, it lacks a very important factor – testing environment. The program was designed for one human player to be able to play against *bots* with graphical interface. Next version of the PA is promised to include a testing environment, but meanwhile *bot* developers had to find a way around this problem.

The following technique for testing Jsbot was used: table was loaded with 9 *bots* and human player *stack size* is set to zero. Hence, bots can play with each other as though the human player is not sitting in the table. Another problem arouse while testing, the fact that *chips* eventually would belong to one player and other players would be sitting without money in the table. For getting around this problem, the bankrolls had to be reset for all

players, excluding the human player. A scripting language called AutoIt [23] was used to do that job after certain intervals. The time between each bankroll reset depended of the amount of Jsbots sitting in the table. 60 seconds for one Jsbot, 120 for two etc. Approximately 10000 games could be tested within 24h, depending on the number of Jsbots and the total number of players, with higher number of Jsbots ~5000 games ran within 24h.

5 Testing methods

Poker is a game with high variance, especially no-limit poker. For this reason identifying the better player is not a trivial task. The profits of players are very dependant on the luck factor. Morgan Hugh Kan demonstrated equal strategies playing against each other 100 000 games of poker [17]. Looking at the profits of both strategies, suggests that one is superior of the other strategy. Though the difference is quite small, only 0.026sb/hand (*small blinds* per hand), the difference still exists. Hence, different testing methods are needed when two strategies are equal to each other. In case of Jsbot testing the performance of *bots* was quite different, so special testing techniques were not used. Though the testing results roughly show which strategy is better than the other, the results cannot be used to precisely bring out each strategy profit in average.

A review of different testing methods that has been used in history of poker testing is in section 5.1. Conditions and methods for testing Jsbot performance are described in section 5.2

5.1 History in testing poker algorithms

Significant study in testing poker players' performance is done in University of Alberta (UA). EVAT and LFAT tools were developed by Darse Billing in his early work on poker and UA best and latest measurement tool, called DIVAT is presented by Darse Billings and Morgan Kan [18] in 2006. Another approach to level out variance and luck factor is using special playing systems, which are described in upcoming paragraphs.

A good example of a special system is duplicate tournament system. It uses independent tournaments with the same series of cards, shuffling the players to different positions for each replay, as described in [20]. Computer programs can replay the games with no memory, so the luck of cards is decreased to some level. But the problem still remains,

because players' strategies deviate from the previous playing history. So even though cards remain the same, the player actions are different and the whole match is different. Duplicate system for *cash games* can also be applied to computer players, with the same problem remaining.

The Expected Value Assessment Tool (EVAT) is a hindsight analysis tool. The actions of a player are judged by whether each action was correct if the opponents' hole cards are revealed. The player action gets score from the difference between the ideal action (when cards are revealed) and the action made by the player. Basically the EVAT tool is a tool for assessing David Sklansky's Fundamental Theorem of Poker [12]. The problem with this measurement is that players are punished even when making correct play. For example, *raising* with second best *hand* gets low score when opponent is holding the best *hand*. On the same time, high score is got in the same situation if opponent is holding third best or worse *hand*. It is almost impossible to conclude so little difference in opponent hand strength in real game situation; so *raising* with second best hand is considered correct play in real play.

The Luck Filtering Analysis Tool (LFAT) is a complementary system for addressing some of the short-comings of the EVAT. Using the same methods to compute the expected value of each situation, the LFAT compares each player's *pot equities* before and after each chance event (*i.e. flop* or *pocket cards* being dealt). Thus the LFAT analysis occurs between the betting rounds, while EVAT occurs between the chance events. The analysis is split into the natural alternating phases of chance outcomes and player actions. LFAT analyzes the outcomes of stochastic elements of poker, without looking at players decisions. In contrast, EVAT considers only player *betting* decisions and disregards luck factor. When two methods work together, big amount of variance is reduced from the game, but treating those two game parts separately introduces some degree of error.

The Ignorant Value Assessment Tool (DIVAT) is also a hindsight tool, but uses decisions from game-theoretic strategy. It is achieved by removing a lot of context from the game to

simplify the decisions, though enough information is considered to make reasonable conclusions for each play. This assessment tool is quite accurate, but modification is needed to support other game types. On Figure 3 you can see bankroll and Divat score of a poker program, from [19]. The match on the figure is between professional poker player Phil Laak and University of Alberta poker program Poki-X. Bankroll and DIVAT Difference line show that the human player was better, just variance is reduced. More interesting is the score difference around games 60-70, the human player had more *bankroll*, but DIVAT score was better for Poki-X.

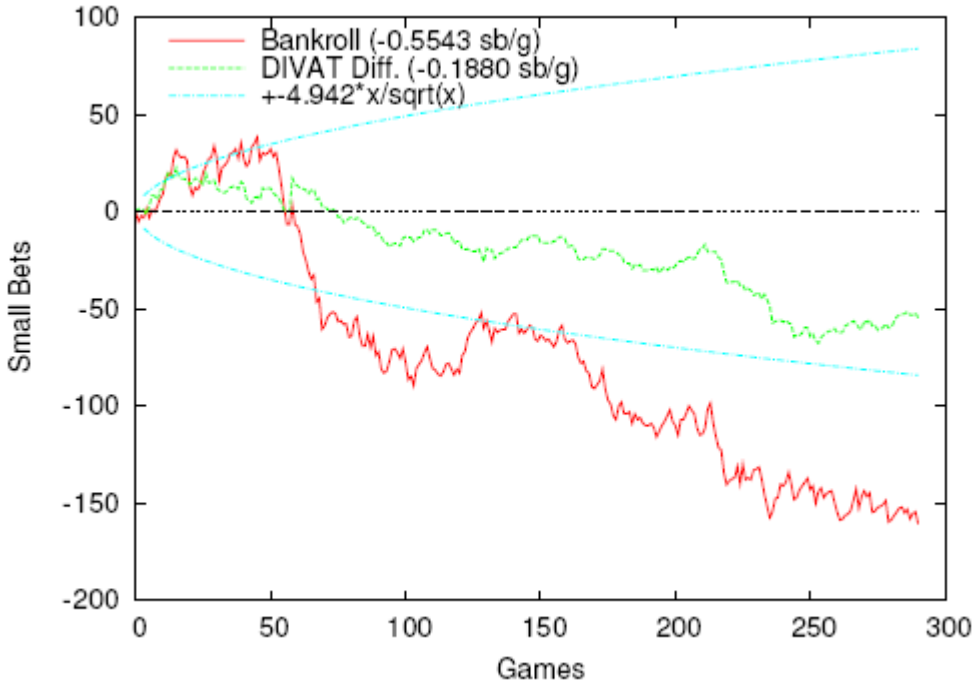

Figure 3. Poki-X bankroll and DIVAT score. Source: [19]

5.2 Test methods used for Jsbot

Jsbot poker games are played in a program Poker Academy Pro (version 2.5) developed by BioTools Incorporated. This is a graphical tool for poker players to improve their poker

skill and runs on Windows. Human players can use it to play against variety of University of Alberta poker algorithms and see statistics of their plays. Poker Academy Pro also has an API for connecting your own bots to the game. This tool was chosen for available opponents to test against.

Though such methods as EVAT, LFAT and DIVAT exist, Jsbot is not tested with those. This is because of extended study and work needed to use those methods and a rough estimation of Jsbot performance can be made by playing enough games.

Jsbot bot is tested against multiple types of poker *bots*. One test run consisted of 10 000 games, which ran approximately 24h.

Jsbot was tested against the following strategies:

AveryBot – Description from Poker Academy: „AveryBot plays an aggressive strategy for No-Limit tournaments. It attempts to build its stack gradually as the blind levels progress while avoiding marginal situations that put a major portion of its stack at risk. The bot is written by Brian Edwards."

ReRaiser – Playing aggressively, if it holds good cards, then always makes a raise (or re-raise if raise was made before). Also likes to raise pre-flop to cut down the field of players.

OddBot – Player who makes deliberately weird plays from time to time.

TournyBot – Bot that follows Darse Billings' no-limit strategy set forth in "A Primer for Playing No-Limit Hold'em Tournaments"

Sklansky – Player who plays according to David Sklansky tournament strategy. The bot either *folds* or goes *all-in pre-flop*, depending on the formula.

PKRoomBot – Tight player, playing „Positive EV Strategy" from online poker website: http://www.pokerroom.com

Solid – Tight player who plays only good hands and almost never bluffs.

6 Results

Jsbot was tested in 16 different situations to bring out the performance from many angles. In all tests 10 000 games were played; where possible games were played in both cooperation and non-cooperation mode. Games were played with each player having *stack* of $100. $1 and $0.5 were *big* and *small blind*. Each player *stack* was reset to $100 after couple of games and profit was summed together.

For each test the average profit per game for each player is brought out. When more than one Jsbot involved in a test, then the important result is the average profit of Jsbots per game, not the best result of Jsbot. The average profit of Jsbots per game is also brought out separately for understanding test results faster.

The average profit per game is brought out as *small blinds* per *hand* (sb/hand). For example, a *bot* gains $4500 in 10 000 games, the average profit would be +0.90sb/hand. A good comparison for the sb/hand is a strategy when a player always *folds*. For example in a ten player game the always folding strategy average profit would be -0.30sb/hand. The exact opponents and the folding strategy average profit are brought out with each test. The description of opponents' strategies is explained in the previous page (Section 5.2).

For bringing out the cooperation factor in tests, same seats were used for all players, when non-cooperation and cooperation modes were tested.

All tests were run under Windows XP operating system, however different hardware were used. The computers used were: two laptops with Mobile AMD Sempron 3000+ processor, one with 1GB, the other with 512MB of ram; One was a virtual machine with 3GB of ram which ran on Tallinn University of Technology server; And the last one was with 64x bit AMD 2200 GHz processor and 2GB of ram. The speed of tests were approximately the

same on all machines, just matches with more than 3 Jsbots present did not run on laptops, due to little free ram.

A summary of all tests are brought out in the Section 6.2.

6.1 Test results

All tests done can be categorized into 4 groups:

1. Full table tests
2. 1 vs. 1 tests
3. Many vs. one tests
4. Self tests

Full table tests consist of games that are played with 9 players. Different number of Jsbots was tested in this environment. Test results and more details can be found in section 6.1.1. *1 vs. 1 tests* are matches between two players. Jsbot was tested against different opponent types, more details are in section 6.1.2. *Many vs. one tests* consist of game series where multiple Jsbots play in cooperation against one opponents, more details are in section 6.1.3. *Self tests* are games where Jsbots competed in cooperation against other Jsbots, who did not play in cooperation. Results and details about *self tests* are in section 6.1.4.

6.1.1 Full table tests

Four different series were played with 9 players. In two of them 3 Jsbots were present, in the other two 2 Jsbots were present. Both cooperation and no-cooperation strategies were tested to see the difference of the strategies in full table poker. With total number of 9 players, it means that a player who folds each *hand* has -0.33 sb/hand value.

The opponents for 3 Jsbot game were: OddBot, PKRoomBot, ReRaiser, Sklansky, Solid and TournyBot. Test results are seen in tables Table 2 and Table 3.

Table 2. Jsbot vs. 6 opponents, non-cooperation

Player	SB/HAND
Solid	+0.39
TournyBot	+0.34
ReRaiser	+0.28
Sklansky	+0.05
PKRoomBot	-0.04
OddBot	-0.14
Jsbot1	-0.22
Jsbot2	-0.28
Jsbot3	-0.41
Jsbots average	**-0.30**

Source: test results.

Table 3. Jsbot vs. 6 opponents, cooperation

Player	SB/HAND
TournyBot	+0.32
Solid	+0.28
ReRaiser	+0.12
PKRoomBot	+0.02
Jsbot1	+0.01
Sklansky	-0.03
OddBot	-0.07
Jsbot2	-0.32
Jsbot3	-0.34
Jsbots average	**-0.22**

Source: test results.

Average profit for 3 Jsbot non-cooperating game is -0.30sb/hand, which is slightly better than always fold strategy; for cooperating game the profit is -0.22sb/hand. From cooperation the average gain for the Jsbots is 0.08sb/hand. So Jsbots lost ~26% less money in cooperation, than in non-cooperation mode.

The opponents for 2 Jsbot game were: AveryBot, OddBot, PKRoomBot, ReRaiser, Sklansky, Solid and TournyBot. Test results are in tables Table 4 and Table 5.

Table 4. 2x Jsbots vs. 7x opponents, no cooperation

Player	SB/HAND
Solid	+0.47
TournyBot	+0.25
ReRaiser	+0.16
OddBot	+0.09
PKRoomBot	+0.02
Sklansky	-0.04
Jsbot2	-0.09
AveryBot	-0.36
Jsbot1	-0.54
Jsbots average	**-0.29**

Source: test results.

Table 5. 2x Jsbots vs. 7x opponents, cooperation

Player	SB/HAND
ReRaiser	+0.21
OddBot	+0.14
PKRoomBot	+0.14
TournyBot	+0.09
Solid	+0.04
Sklansky	-0.04
Jsbot2	-0.11
AveryBot	-0.15
Jsbot1	-0.33
Jsbots average	**-0.22**

Source: test results.

Average sb/hand for 2 Jsbot non-cooperating game is -0.29, which is quite close to game where 3 Jsbots were present; for cooperating game it is -0.22. From cooperation the average gain for the Jsbots is 0.07sb/hand. So Jsbots lost ~24% less money when they made cooperation, also quite similar to 3 Jsbot game.

6.1.2 1 vs. 1 tests

Three game series were played during 1 vs. 1 testing. *Tight, aggressive and very tight* players were the opponents. These tests show the capability of Jsbot in 1 vs. 1 situation and

also brings out if the strategy itself is reasonable or not. Cooperation could not be tested in those circumstances. For two player game, the always fold strategy profit is -1.5 sb/hand.

For tight opponent Solid was chosen, because it was one of the tightest players in the previous testing rounds and showed good results. Test results can be seen in Table 6.

Table 6. Jsbot vs. a tight opponent (Solid)

Player	**SB/HAND**
Solid	+0.73
Jsbot	-0.73

Source: test results.

For aggressive opponent ReRaiser was chosen, because of its playing style and also good performance. AveryBot was another possible aggressive opponent, but with not so good performance in full table tests. Test results are in Table 7.

Table 7. Jsbot vs. an aggressive opponent (ReRaiser)

Player	**SB/HAND**
ReRaiser	+0.75
Jsbot	-0.75

Source: test results.

For very tight opponent Sklansky was chosen, because of its playing strategy. Sklansky goes *all-in* or *folds* with each hand in *pre-flop* round. Effectively no decisions are made in *flop*, *turn* or *river* betting rounds, making it an interesting opponent to test against. Test results are in Table 8.

Table 8. Jsbot vs. a very tight opponent (Sklansky)

Player	**SB/HAND**
Sklansky	+0.22
Jsbot	-0.22

Source: test results.

For both first tests made, the sb/hand value is close to -0.75, which is exactly two times better than always fold strategy. Against Sklansky, the profit is much better: -0.22 sb/hand. Even though the results are better than in full table poker, the negative profit still confirms that Jsbot algorithm have to be dramatically improved.

6.1.3 Many vs. one tests

In *many vs. one tests* eight series were played. In four of them 3 Jsbots and in the other four 5 Jsbots were used. A tight and aggressive style was chosen for opponents to have test results against different strategies. Both cooperation and no-cooperation strategies were tested. For tight player, Solid was the opponent again and ReRaiser as aggressive opponent.

When 5 Jsbots competed against one opponent, the always fold strategy profit is -0.5 sb/hand. The results of cooperation strategy against Solid can be seen Table 9 and no-cooperation strategy results are in Table 10. Test results of 5 Jsbots against ReRaiser can be seen in tables 11 and 12.

Table 9. 5x Jsbots vs. Solid, cooperation

Player	**SB/HAND**
Jsbot5	+1.42
Jsbot4	+0.36
Solid	+0.13
Jsbot1	-0.43
Jsbot2	-0.50
Jsbot3	-0.97
Jsbots average	**-0.02**

Source: test results

Table 10. 5x Jsbots vs. Solid, no cooperation

Player	**SB/HAND**
Jsbot2	+0.38

Jsbot3	+0.25
Solid	+0.05
Jsbot5	-0.17
Jsbot4	-0.25
Jsbot1	-0.26
Jsbots average	**-0.01**

Source: test results

Table 11. 5x Jsbots vs. ReRaiser, cooperation

Player	**SB/HAND**
Jsbot5	+1.16
ReRaiser	+0.23
Jsbot4	+0.22
Jsbot3	-0.26
Jsbot2	-0.61
Jsbot1	-0.74
Jsbots average	**-0.04**

Source: test results

Table 12. 5x Jsbots vs. ReRaiser, no cooperation

Player	**SB/HAND**
Jsbot1	+0.29
Jsbot4	+0.26
Jsbot5	+0.08
ReRaiser	-0.11
Jsbot2	-0.23
Jsbot3	-0.29
Jsbots average	**+0.02**

Source: test results

The results show clearly that in this game, cooperation strategy used by Jsbots is backfiring. The expected result is that cooperation increases profits, but in reality it decreases them. In case of tight opponent, the average profit of Jsbots dropped from -0.01 sb/hand to -0.02sb/hand and against aggressive opponent the fall is even more dramatic, from +0.02sb/hand to -0.04sb/hand. This failed cooperation can be explained by the cooperation strategy Jsbots use. They do not play against other Jsbots and effectively play 1 vs. 1 game against the lonely opponent in the table. In case the lonely opponent is one of the last ones to act, Jsbots *bet* or *raise* as it would be sensible in 1vs1 game. But in 6 player

game it is not reasonable because the lonely opponent can wait for better hand, only paying 0.5 *small blinds* in average every game.

Also series of *3 vs. 1* games were run. In this format the aggressive cooperation strategy of Jsbots paid off and resulted in profit gain. Again Solid and ReRaiser were chosen as opponents. The results against Solid are in tables Table 13 and Table 14; the results against ReRaiser are in tables Table 15 and Table 16. The always folding strategy profit in 4 player games is -0.75 sb/hand.

Table 13. 3x Jsbots vs. Solid, cooperation

Player	**SB/HAND**
Jsbot3	+1.66
Solid	+0.10
Jsbot2	-0.81
Jsbot1	-0.95
Jsbots average	**-0.03**

Source: test results

Table 14. 3x Jsbots vs. Solid, no cooperation

Player	**SB/HAND**
Jsbot1	+0.51
Solid	+0.22
Jsbot2	-0.33
Jsbot3	-0.40
Jsbots average	**-0.07**

Source: test results

Table 15. 3x Jsbots vs. ReRaiser, cooperation

Player	**SB/HAND**
Jsbot3	+1.30
Jsbot2	+0.06
ReRaiser	-0.04
Jsbot1	-1.32
Jsbots average	**+0.01**

Source: test results

Table 16. 3x Jsbots vs. ReRaiser, no cooperation

Player	SB/HAND
Jsbot3	+0.62
ReRaiser	+0.17
Jsbot2	+0.03
Jsbot1	-0.82
Jsbots average	**-0.06**

Source: test results

Results show that in three versus one game, the cooperation strategy is successful. Although Jsbots plays very aggressively against a lonely opponent, the results show that in 4 player game it is justified. Against the tight opponent Jsbots average profit raised from -0.07sb/hand to -0.03sb/hand and against the aggressive opponent raise was from -0.06sb /hand to +0.01sb/hand.

In conclusion many vs. one tests showed interesting aspect of cooperation strategy, where players competed against single opponent. As the results show, cooperation strategy that works in full-table games may not be sufficient in many vs. one game situation.

6.1.4 Self tests

Self tests are made when testing the performance of differences in strategies. For Jsbot cooperation and no-cooperation strategy was tested. While self tests give some sort of measurement for the cooperation, it is one of the worst measurements available. This is because the diversity of strategies is absent, unlike real game situations where players with many different strategies compete against each other. Only one test was conducted, because the measurement does not hold great value. It only shows the performance difference of cooperating Jsbots when playing against non-cooperating Jsbots. Series of 2 vs. 2 game was played, with always fold strategy having -0.75 sb/hand profit. Jsbot1 and Jsbot2 did not play cooperating strategy, while Jsbot1-coop and Jsbot2-coop cooperated with each other. Test results can be seen in Table 17.

Table 17. 2 vs. 2 Jsbot match

Player	SB/HAND
Jsbot1	+0.99
Jsbot1-coop	+0.66
Jsbot2-coop	-0.19
Jsbot2	-1.46
Cooperating Jsbots average	**+0.24**

Source: test results

Cooperating Jsbots had average profit of +0.24sb/hand, which shows that cooperation against similar opponents is very effective. This result proves that cooperation is successful, but when developing cooperation strategy further, other types of tests should be conducted to show overall performance.

6.2 Summary of results

Totally 16 series of games were played, making 160 000 games in total. The games when Jsbots did not use cooperation showed clearly that Jsbot needs further development in becoming a reasonable poker AI system. Many tests ended with profit that is close to always *fold* strategy, with exception to 1 vs. 1 tests. Although the results of 1 vs. 1 tests were better when compared to always *fold* strategy (Jsbots lost 2 times less than always *fold* strategy), Jsbots lost to all opponents and cannot be considered a reasonable poker AI system.

Despite the poor performance of Jsbot, results from cooperating are encouraging for further study in cooperation possibilities in poker. Average gain per Jsbot from cooperation was around 0.07-0.08 sb/hand in full table games and even higher in many vs. one tests. Though many vs. one tests had profit gain of 0.04 (against tight opponent) and 0.07sb/hand (against aggressive opponent) in 3 vs. 1 tests, the cooperation strategy decreased the average profit in 5 vs. 1 games 0.01 (against tight opponent) and 0.07sb/hand (against aggressive opponent). These results show clearly that cooperation in poker is not a trivial task and depends much from the game situation as non-cooperating poker strategy does.

Summary

Jsbot uses player modeling and simple algorithm for calculating expected value for possible 7 moves. The expected value for each move is calculated without considering future betting rounds and without considering possibility that opponents might raise. The test results show clearly that this simplistic value calculation is not enough for reasonable poker AI system, though it is enough to offer some competition to other poker strategies. Possible improvements to the Jsbot strategy is to consider more possible opponent moves, allowing Jsbot to harness such strategy as *slow playing* and calculate expected value more accurately. Considering more future actions is not an easy task, because poker game tree grows very fast, but it can be done by optimizing algorithms used. Even looking at one or two future actions for each player should result in a more accurate estimation of expected value.

When Jsbots are cooperating, then they share hidden *pocket cards* and do not play against each other. This cooperation strategy proved to be quite valuable in full table poker and in 4 player game, where 3 Jsbots competed against one opponent. Those good results give reason for studying cooperation in poker in more depth. Improvements can also be made to the cooperation strategy so that Jsbots "agree" on one strategy and follow it. As results from 5 vs. 1 tests show, without a unified strategy the cooperation in poker may even be invaluable. The unified strategy would be also interesting in full table poker, where it might increase average profit of cooperating players even more.

Testing the further developments of Jsbot non-cooperating and cooperating strategies is quite difficult as one test of 10 000 games runs about 24h, effectively making the testing of new features' performance a slow process. For fastening poker testing a different testing environment should be used and a tool like DIVAT should be modified for no-limit poker.

Resümee

Lõputöö peamiseks eesmärgiks on lihtsa iseõppiva pokkerimängija programmi loomine. Lisaks võimaldab see tehisintellekt teha reeglitevastast koostööd teiste sarnaste pokkerimängu programmidega. Lõputöös analüüsitakse sellise koostöö mõju pokkeriprogrammide edukusele.

Üks probleemidest lõputöö tegemisel oli tehisintellekti süsteemi loomine, mis suudab pokkerimängus vastaste strateegiat õppida ja vastavalt sellele oma mängu strateegiat muuta. Samuti kuulusid probleemide hulka koostöö strateegia loomine ning põhjaliku testimise läbiviimine erinevate mängijate vastu varieeruvates olukordades.

Lõputöö tulemuseks on tehisintellekti süsteem, mis mängib pokkerit õppides vastaste mängu strateegiat ja suurendab enda kasumit tehes koostööd teiste sarnaste süsteemidega.

Literature

[1] D. Billings, D. Papp, J. Schaeffer, D. Szafron. “Poker as a Testbed for Machine Intelligence Research”. 06.10.1998. University of Alberta
04.09.2007 <http://www.cs.ualberta.ca/~jonathan/Papers/Papers/ai98.poker.html>

[2] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. “The Challenge of Poker”. Artificial Intelligence Journal vol. 134 01.2002: 201-240

[3] D. Koller, A. Pfeffer. “Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)”. Montreal, Canada: Morgan Kaufmann, 1995

[4] J.F. Nash. “Non-cooperative games”. Annals of Mathematics vol. 54 11.1951: 286-295

[5] J. F. Nash, L. S. Shapley. “A Simple three-person poker game”. Contributions to the Theory of Games vol. 1 06.1950: 105-116

[6] J. von Neumann, O. Morgenstern. “The Theory of Games and Economic Behaviour”. Princton, USA: Princeton University Press, 1947.

[7] “The First Man-Machine Poker Championship”. 06.2007. University of Alberta Computer Poker Research Group.
04.09.2007 <http://www.cs.ualberta.ca/~games/poker/man-machine/>

[8] B. Clark. “The Dying Days of Las Vegas 1-5 Stud”. USA: Two Plus Two Publishing, 2006

[9] D. Brunson. “Super/System: A course in power poker”. USA: B&G Publishing Company, 1978

[10] D. Sklansky. “Sklansky on Poker”. USA: Two Plus Two Publishing, 1994

[11] D. Sklansky. “Hold'em Poker”. USA: Two Plus Two Publishing, 1994

[12] D. Sklansky. “Theory of Poker”. USA: Two Plus Two Publishing, 1994

[13] D. Sklansky, Mason Malmuth. “Hold'em Poker for Advanced Players, 21st Century Edition”. USA: Two Plus Two Publishing, 1999

[14] L. Jones. “A Glossary of Poker Terms”. 11.09.1994. ConJelCo.
04.09.2007 <http://www.conjelco.com/pokglossary.html>

[15] H. W. Kuhn. “A simplified two-person poker”. Contributions to the Theory of Games vol. 1 06.1950: 97-103

[16] R. Andersson. “Pseudo-Optimal Strategies in No-Limit Poker”.Master's thesis. Department of Computing Science, Umeå University. 08.05.2006. Umeå, Sweden.

[17] M.H. Kan. “Postgame Analysis of Poker Decisions”. Master's thesis. Department of Computing Science, University of Alberta. 01.2007. Alberta, Canada.

[18] A. Davidson. “Opponent modeling in poker”. Master's thesis, Department of Computing Science, University of Alberta, 2002. Alberta, Canada.

[19] D. Billings, M.H. Kan. “A Tool for the Direct Assessment of Poker Decisions”. 10.2006. University of Alberta
04.09.2007 <http://www.cs.ualberta.ca/~darse/Papers/divat-icgaj.html>

[20] D. Billings. “Computer Poker”. 30.10.1995. University of Alberta
04.09.2007 <http://www.cs.ualberta.ca/~darse/msc-essay/thesis.html>

[21] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu. “An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions”. Journal of the ACM vol. 45 no. 6 11.1998: 891-923

[22] “Poker Academy - Our Poker Community”. BioTools Incorporated.
04.09.2007 <http://www.poker-academy.com/community.php>

[23] “AutoIt Script Home Page”. J. Bennett. 04.09.2007 <http://www.autoitscript.com/>

[24] “Poker Academy - Your Source for Great Poker Software”. BioTools Incorporated
04.09.2007 <http://www.poker-academy.com/>

[25] B. Arkin, F. Hill, S. Marks, M. Schmid, T.J. Walls “|How We Learned to Cheat at Online Poker: A Study in Software Security”. Cigital. 09.1999
04.09.2007 <http://www.cigital.com/papers/download/developer_gambling.php>

List of figures

List of formulas

List of tables

Appendix 1. Poker card abbreviations

There are 52 cards in a poker deck. Writing the cards out in long version (i.e. Ace of Spades) takes much room and abbreviations are used in most poker literature. Here is explanation of abbreviations used in this paper. Poker card consists of two parts: strength (from 2 to Ace) and suit (from Hearts to Spades).

Possible strengths are:
A – Ace
K – King
Q – Queen
J – Jack
T – Ten
9 – Nine
8 – Eight
7 – Seven
6 – Six
5 – Five
4 – Four
3 – Three
2 – Two

Possible suits are:
s – Spade
c – Club
d – Diamond
h – Heart

So Ten of Spades is Ts; Ace of Hearts is Ah.

Appendix 2. Poker glossary

The glossary of abbreviations and terminology is taken from [14], only relevant terms are brought out in this listing.

Action

(1) Opportunity to act. If a player appears not to realize it's his turn, the dealer will say "Your action, sir."

(2) Bets and raises. "If a third heart hits the board and there's a lot of action, you have to assume that somebody has made the flush."

All-In

To run out of chips while betting or calling. In table stakes games, a player may not go into his pocket for more money during a hand. If he runs out, a side pot is created in which he has no interest. However, he can still win the pot for which he had the chips. Example: "Poor Bob. He made quads against the big full house, but he was all-in on the second bet."

Bet

An action when a player puts money into pot. The other players need to call that sum or raise it, to continue playing. See also Raise.

Big Blind

The larger of the two blinds typically used in a hold'em game. The big blind is a full first round bet. See also "blind" and "small blind."

Blind

A forced bet (or partial bet) put in by one or more players before any cards are dealt. Typically, blinds are put in by players immediately to the left of the button. See also "live blind."

Board

All the community cards in a hold'em game -- the flop, turn, and river cards together. Example: "There wasn't a single heart on the board."

Bot

Short for "robot". In a poker context, a program that plays poker online with no (or minimal) human intervention.

Button

A white acrylic disk that indicates the (nominal) dealer. Also used to refer to the player on the button. Example: "Oh, the button raised."

Call

To put into the pot an amount of money equal to the most recent bet or raise. The term "see" (as in "I'll see that bet") is considered colloquial.

Check

(1) To not bet, with the option to call or raise later in the betting round. Equivalent to betting zero dollars. (2) Another word for chip, as in poker chip.

Draw

To play a hand that is not yet good, but could become so if the right cards come. Example: "I'm not there yet -- I'm drawing." Also used as a noun. Example: "I have to call because I have a good draw."

Flop

The first three community cards, put out face up, all together.

Free Card

A turn or river card on which you don't have to call a bet because of play earlier in the hand (or because of your reputation with your opponents). For instance, if you are on the button and raise when you flop a flush draw, your opponents may check to you on the turn. If you make your flush on the turn, you can bet. If you don't get it on the turn, you can check as well, seeing the river card for "free."

Hit

As in "the flop hit me," meaning the flop contains cards that help your hand. If you have AK, and the flop comes K-7-2, it hit you.

Implied Odds

Pot odds that do not exist at the moment, but may be included in your calculations because of bets you expect to win if you hit your hand. For instance, you might call with a flush draw on the turn even though the pot isn't offering you quite 4:1 odds (your chance of making the flush) because you're sure you can win a bet from your opponent on the river if you make your flush.

Kicker

An unpaired card used to determine the better of two near-equivalent hands. For instance, suppose you have AK and your opponent has AQ. If the flop has an ace in it, you both have a pair of aces, but you have a king kicker. Kickers can be vitally important in hold'em.

Muck

The pile of folded and burned cards in front of the dealer. Example: "His hand hit the muck so the dealer ruled it folded even though the guy wanted to get his cards back." Also used as a verb. Example: "He didn't have any outs so he mucked his hand."

No-Limit

A version of poker in which a player may bet any amount of chips (up to the number in front of him) whenever it is his turn to act. It is a very different game from limit poker.

Nuts

The best possible hand given the board. If the board is Ks-Jd-Ts-4s-2h, then As-Xs is the nuts. You will occasionally hear the term applied to the best possible hand of a certain category, even though it isn't the overall nuts. For the above example, somebody with Ah-Qc might say they had the "nut straight."

Pocket

Your unique cards that only you can see. For instance, "He had pocket sixes" (a pair of sixes), or "I had ace-king in the pocket."

Pocket Pair

A hold'em starting hand with two cards of the same rank, making a pair. Example: "I had big pocket pairs seven times in the first hour. What else can you ask for?"

Pot-Limit

A version of poker in which a player may bet up to the amount of money in the pot whenever it is his turn to act. Like no-limit, this is a very different game from limit poker.

Pot Odds

The amount of money in the pot compared to the amount you must put in the pot to continue playing. For example, suppose there is $60 in the pot. Somebody bets $6, so the pot now contains $66. It costs you $6 to call, so your pot odds are 11:1. If your chance of having the best hand is at least 1 out of 12, you should call. Pot odds also apply to draws. For instance, suppose you have a draw to the nut flush with one card left to come. In this case, you are about a 4:1 underdog to make your flush. If it costs you $8 to call the bet, then there must be about $32 in the pot (including the most recent bet) to make your call correct.

Raise

An action when a player puts more money into pot than was put with previous bet. Raise always comes after a bet has been made. When a player put money into pot, even though he might have checked, is considered a bet.

River

The fifth and final community card, put out face up, by itself. Also known as "fifth street." Metaphors involving the river are some of poker's most treasured cliches, e.g., "He drowned in the river."

Second Pair

A pair with the second highest card on the flop. If you have As-Ts, and the flop comes Kd-Th-6c, you have flopped second pair. See "top pair."

Semi-Bluff

A powerful concept first discussed by David Sklansky. It is a bet or raise that you hope will not be called, but you have some outs if it is. A semi-bluff may be correct when betting for value is not correct, a pure bluff is not correct, but the combination of the two may be a positive expectation play. Example: you have Ks-Qs, and the flop is Th-5s-Jc. If you bet now, it's a semi-bluff. You probably don't have the best hand, and you'd like to see your opponents fold immediately. Nevertheless, if you do get callers, you could still improve to the best hand.

Set

Three of a kind when you have two ofthe rank in your hand, and there is one on the board.

Short Stack

A number of chips that is not very many compared to the other players at the table. If you have $10 in front of you, and everybody else at the table has over $100, you are playing on a short stack.

Showdown

The point at which all players remaining in the hand turn their cards over and determine who has the best hand -- i.e., after the fourth round of betting is completed. Of course, if a final bet or raise is not called, there is no showdown.

Slow Play

To play a strong hand weakly so more players will stay in the pot.

Small Blind

The smaller of two blind bets typically used in a hold'em game. Normally, the small blind is one-third to two-thirds of a first round bet. See also "big blind" and "blind."

Suited

A hold'em starting hand in which the two cards are the same suit. Example: "I had to play J-3 -- it was suited."

Tell

A clue or hint that a player unknowingly gives about the strength of his hand, his next action, etc. May originally be from "telegraph" or the obvious use that he "tells" you what he's going to do before he does it.

Top Pair

A pair with the highest card on the flop. If you have As-Qs, and the flop comes Qd-Th-6c, you have flopped top pair. See "second pair."

Top Set

The highest possible trips. Example: you have Tc-Ts, and the flop comes Td-8c-9h. You have flopped top set.

Turn

The fourth community card. Put out face up, by itself. Also known as "fourth street."

Under the Gun

The position of the player who acts first on a betting round. For instance, if you are one to the left of the big blind, you are under the gun before the flop.

Value

As in "bet for value". This means that you would actually like your opponents to call your bet (as opposed to a bluff). Generally it's because you have the best hand. However, it can also be a draw that, given enough callers, has a positive expectation.

Variance

A measure of the up and down swings your bankroll goes through. Variance is not necessarily a measure of how well you play. However, the higher your variance, the wider swings you'll see in your bankroll.